Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Современные системы программирования»

Тема: «Таблицы»

Выполнил:

Студент 4 курса

Группы ИИ-16

Журавлёв В.А.

Проверил:

Коренкович О. Г.

Брест 2021

**Цель работы:** освоить работу с двумерными массивами; познакомиться с классом StringGrid; научиться программировать алгоритмы поиска наибольшего и наименьшего значения, научиться обрабатывать исключительные ситуации.

**Ход работы**

**Задание:**

Создать программу, выполняющую:

– заполнение таблицы случайными целыми числами в диапазоне

от -20 до 20;

– возможность изменения данных вручную;

– нахождение сумм, средних, минимальных и максимальных значений по строкам, столбцам и таблице в целом с заполнением таблицы результатов;

- загрузка из xml-файла в таблицу данных документа.

**Код программы:**

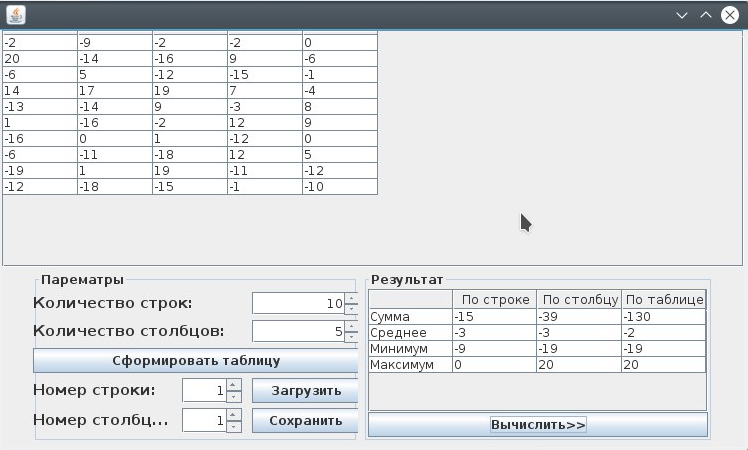
public class Start {  
 public static void main(String[] args) {  
 java.awt.EventQueue.invokeLater(() -> new TableFrame().setVisible(true));  
 }  
}

package example;  
  
import javax.swing.table.AbstractTableModel;  
import java.util.Arrays;  
  
public class ValuesTM extends AbstractTableModel {  
 String[] colNames;  
 String[][] colValues;  
  
 public String[][] getValues(){  
 return this.colValues;  
 }  
  
 public ValuesTM(String[][] values) {  
 this.colNames = new String[values[0].length];  
 this.colValues = Arrays.copyOf(values,values.length);  
 }  
  
 public ValuesTM(String[] cols, String[][] values) {  
 this.colNames = Arrays.copyOf(cols,cols.length);  
 this.colValues = Arrays.copyOf(values,values.length);  
 }  
  
 public ValuesTM(int rowsN, int colsN) {  
 this.colNames = new String[colsN];  
 this.colValues = new String[rowsN][colsN];  
  
 for (int row = 0; row < rowsN; row++) {  
 colValues[row] = new String[colsN];  
 for (int col = 0; col < colsN; col++) {  
 colValues[row][col] = "";  
 }  
 }  
 }  
  
 @Override  
 public String getColumnName(int column) {  
 return colNames[column];  
 }  
  
 @Override  
 public int getRowCount() {  
 return colValues.length;  
 }  
  
 @Override  
 public int getColumnCount() {  
 return colNames.length;  
 }  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 if (colValues.length >= rowIndex && columnIndex <= colNames.length) {  
 return colValues[rowIndex][columnIndex]==null?"":colValues[rowIndex][columnIndex];  
 }  
 return "";  
 }  
  
 @Override  
 public void setValueAt(Object aValue, int rowIndex, int columnIndex) {  
 if (colValues.length >= rowIndex && columnIndex <= colNames.length) {  
 colValues[rowIndex][columnIndex] = String.valueOf(aValue);  
 }  
 }  
  
 @Override  
 public boolean isCellEditable(int rowIndex, int columnIndex) {  
 return true;  
 }  
}

package example;  
  
import javax.swing.table.AbstractTableModel;  
  
public class ResultTM extends AbstractTableModel {  
 String[] colNames = {"", "По строке", "По столбцу", "По таблице"};  
 String[][] colValues = {  
 {"Cумма", "", "", ""},  
 {"Среднее", "", "", ""},  
 {"Минимум", "", "", ""},  
 {"Максимум", "", "", ""}  
 };  
  
 public ResultTM() {  
 }  
  
 public void setResultSumm(String byTableRow, String byTableCol, String byTable){  
 this.colValues[0][1]=byTableRow;  
 this.colValues[0][2]=byTableCol;  
 this.colValues[0][3]=byTable;  
 }  
 public void setResultAverage(String byTableRow, String byTableCol, String byTable){  
 this.colValues[1][1]=byTableRow;  
 this.colValues[1][2]=byTableCol;  
 this.colValues[1][3]=byTable;  
 }  
 public void setResultMinimum(String byTableRow, String byTableCol, String byTable){  
 this.colValues[2][1]=byTableRow;  
 this.colValues[2][2]=byTableCol;  
 this.colValues[2][3]=byTable;  
 }  
 public void setResultMaximum(String byTableRow, String byTableCol, String byTable){  
 this.colValues[3][1]=byTableRow;  
 this.colValues[3][2]=byTableCol;  
 this.colValues[3][3]=byTable;  
 }  
  
 @Override  
 public String getColumnName(int column) {  
 return colNames[column];  
 }  
  
 @Override  
 public int getRowCount() {  
 return colValues.length;  
 }  
  
 @Override  
 public int getColumnCount() {  
 return colNames.length;  
 }  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 if (colValues.length >= rowIndex && columnIndex <= colNames.length) {  
 return colValues[rowIndex][columnIndex] == null ? "" : colValues[rowIndex][columnIndex];  
 }  
 return "";  
 }  
  
 @Override  
 public boolean isCellEditable(int rowIndex, int columnIndex) {  
 return false;  
 }  
}

public class TableFrame extends JFrame {  
  
 int startVal\_mainRows = 10, // кол-во строк  
 int startVal\_mainCols = 5, // кол-во столбцов  
 int startVal\_minRand = -20, // произв. число минимальное значение  
 int startVal\_maxRand = 20;// произв. число макснимальное значение  
  
 private JScrollPane mainTableSP; // Панель прокрутки для главной таблицы  
 private JTable mainTable; //Главная таблица  
  
 private JPanel bottomPanel; // Нижняя панель с кнопками и таблицей результатов  
  
 // Параметры Инициализации таблицы  
 private JLabel colCountLabel;  
 private JSpinner colCountSelector;  
  
 private JLabel rowCountLabel;  
 private JSpinner rowCountSelector;  
  
 // Кнопка Генерация модели таблицы  
 private JButton generateTM;  
  
 // Параметры расчета по таблице  
 private JLabel colNumberLabel;  
 private JSpinner colNumberSelector;  
  
 private JLabel rowNumberLabel;  
 private JSpinner rowNumberSelector;  
  
 // Кнопка получения результатов  
 private JButton calculate;  
 // Кнопка сохранения в xml  
 private JButton load;  
 private JButton save;  
 private JScrollPane resultTableSP; // Панель прокрутки для таблицы с результатами  
 private JTable resultTable; //Таблица с результатами  
  
 public TableFrame() {  
 this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 this.setLayout(new BorderLayout());  
 this.setSize(750, 450);  
 this.setMinimumSize(getSize());  
 initComponents();  
 }  
  
 private ValuesTableModel initValuesTM(String[][] values) {  
 return new ValuesTableModel(values);  
 }  
  
 private ValuesTableModel initValuesTM(int rowsN, int colsN) {  
 return this.initValuesTM(rowsN, colsN, startVal\_minRand, startVal\_maxRand);  
 }  
  
 private ValuesTableModel initValuesTM(int \_rowsN, int \_colsN, int minRand, int maxRand) {  
  
 int rowsN = Math.max(\_rowsN, 0);  
 int colsN = Math.max(\_colsN, 0);  
  
 String[] colNames = new String[colsN];  
 String[][] colValues = new String[rowsN][colsN];  
  
 Random random = new Random();  
  
 for (int row = 0; row < rowsN; row++) {  
 colValues[row] = new String[colsN];  
 for (int col = 0; col < colsN; col++) {  
 colValues[row][col] = String.valueOf((random.nextInt((maxRand + 1) - minRand) + minRand));  
 }  
 }  
  
 return new ValuesTableModel(colNames, colValues);  
 }  
  
 private void initComponents() {  
  
 mainTableSP = new JScrollPane();  
 mainTable = new JTable(initValuesTM(startVal\_mainRows, startVal\_mainCols, startVal\_minRand, startVal\_maxRand));  
 mainTableSP.setViewportView(mainTable);//создает область просмотра главной таблицы  
 mainTable.getTableHeader().setReorderingAllowed(false); // фиксация табличного пространства (запрет перемещения столбцов) главной таблицы  
  
 mainTable.setShowGrid(true);  
 mainTable.setAutoResizeMode(JTable.AUTO\_RESIZE\_OFF);  
  
  
 ListSelectionModel mainTableSelectionModel = mainTable.getSelectionModel();  
 mainTableSelectionModel.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);  
  
 mainTable.setCellSelectionEnabled(true);  
 mainTable.setRowSelectionAllowed(true);  
 mainTable.setColumnSelectionAllowed(true);  
  
 mainTableSelectionModel.addListSelectionListener(new ListSelectionListener() {  
 public void valueChanged(ListSelectionEvent e) {  
 mainTable\_handleSelectionEvent(e);  
 }  
 });  
  
 mainTable.getColumnModel().getSelectionModel().addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 mainTable\_handleSelectionEvent(e);  
 }  
 });  
  
 this.add(mainTableSP, BorderLayout.CENTER);  
  
 bottomPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 5));  
 bottomPanel.setBorder(BorderFactory.createEmptyBorder(0, 5, 0, 5));  
  
 Dimension bottomPanelDimension = new Dimension(700, 180);  
 bottomPanel.setSize(bottomPanelDimension);  
 bottomPanel.setPreferredSize(bottomPanelDimension);  
 bottomPanel.setMinimumSize(bottomPanelDimension);  
 bottomPanel.setMaximumSize(bottomPanelDimension);  
  
   
 bottomPanel.add(new JPanel() {{  
 rowCountLabel = new JLabel();  
 rowCountSelector = new JSpinner();  
 rowCountSelector.setValue(startVal\_mainRows);  
 colCountLabel = new JLabel();  
 colCountSelector = new JSpinner();  
 colCountSelector.setValue(startVal\_mainCols);  
 generateTM = new JButton();  
 rowNumberLabel = new JLabel();  
 rowNumberSelector = new JSpinner();  
 load = new JButton();  
 colNumberLabel = new JLabel();  
 colNumberSelector = new JSpinner();  
 save = new JButton();  
  
 this.setLayout(new GridBagLayout());  
 this.setBorder(new TitledBorder("Парематры"));  
  
 Dimension optionsPanelDimension = new Dimension(325, bottomPanelDimension.height - 10);  
 this.setSize(optionsPanelDimension);  
 this.setPreferredSize(optionsPanelDimension);  
 this.setMinimumSize(optionsPanelDimension);  
 this.setMaximumSize(optionsPanelDimension);  
  
 ((GridBagLayout) getLayout()).columnWidths = new int[]{130, 70, 100, 0};  
 ((GridBagLayout) getLayout()).rowHeights = new int[]{28, 28, 28, 0, 24, 24};  
  
 //---- rowCountLabel ----  
 rowCountLabel.setText("Количество строк: ");  
 rowCountLabel.setFont(rowCountLabel.getFont().deriveFont(rowCountLabel.getFont().getStyle() | Font.BOLD, rowCountLabel.getFont().getSize() + 3f));  
 add(rowCountLabel, new GridBagConstraints(0, 0, 2, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
 add(rowCountSelector, new GridBagConstraints(2, 0, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
  
 //---- colCountLabel ----  
 colCountLabel.setText("Количество столбцов: ");  
 colCountLabel.setFont(colCountLabel.getFont().deriveFont(colCountLabel.getFont().getStyle() | Font.BOLD, colCountLabel.getFont().getSize() + 3f));  
 add(colCountLabel, new GridBagConstraints(0, 1, 2, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
 add(colCountSelector, new GridBagConstraints(2, 1, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
  
 //---- generateTM ----  
 add(generateTM, new GridBagConstraints(0, 2, 3, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
 generateTM.setAction(new AbstractAction("Сформировать таблицу") {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 generateTM\_handleAction(e);  
 }  
 });  
  
 rowNumberLabel.setText("Номер строки: ");  
 rowNumberLabel.setFont(rowNumberLabel.getFont().deriveFont(rowNumberLabel.getFont().getStyle() | Font.BOLD, rowNumberLabel.getFont().getSize() + 3f));  
 add(rowNumberLabel, new GridBagConstraints(0, 4, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
 add(rowNumberSelector, new GridBagConstraints(1, 4, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
  
 add(load, new GridBagConstraints(2, 4, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
 load.setAction(new AbstractAction("Загрузить") {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 load\_handleAction(e);  
 }  
 });  
  
 colNumberLabel.setText("Номер столбца: ");  
 colNumberLabel.setFont(colNumberLabel.getFont().deriveFont(colNumberLabel.getFont().getStyle() | Font.BOLD, colNumberLabel.getFont().getSize() + 3f));  
 add(colNumberLabel, new GridBagConstraints(0, 5, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 0, 10), 0, 0));  
 add(colNumberSelector, new GridBagConstraints(1, 5, 1, 1, 0.0, 0.0);}  
}

**Результат работы программы:**



**Вывод:** освоил работу с двумерными массивами; познакомился с классом StringGrid; научился программировать алгоритмы поиска наибольшего и наименьшего значения, научился обрабатывать исключительные ситуации.